CS 374 A: Intro to Algorithms & Models of Computation

https://courses.granger.illinois.edu/cs374a21/sp2024

Instructors: Timothy Chan & Ruta Mehta

8 TAs, 22 CAs

Hws: 11 guided problem sets (GPSs) on PrairieLearn (autograded)
+ 11 written homeworks, each with 2 problems
  (may work in groups of ≤3)

→ total = 33 HW problems
  (drop lowest 6)

No late GPS

HW: ≤ 24 hr late with 40% penalty
  (zero afterwards)

Extenuating circumstances e.g. illness ⇒ ask instructor

Exams

Midterm 1: Feb 19 Mon 7p-9p
Midterm 2: Apr 8 Mon 7p-9p (conflict: TBA)
Final: TBA

Grades

HW 28%
Mid 1 21%
Mid 2 21%
Final 30%

Fixed cut-offs for letter-grade
(Curved cut-offs as backup
take better of two)
See web pages

Note: Sections A & B are completely independent

Resources: Jeff’s book + lecture scribbles
To do well: attend all lectures (ask Qs!) & labs
get help during OTHs, platez, etc.

Overview

Introduction to CS theory

Goal 1: how to solve problems (efficiently)
  design algorithms & analyses

Goal 2: how to show that a problem can’t be solved (efficiently)
  mathematically prove

Outline

Part I. Models of Computation
  → finite automata ↔ reg exprs
  context-free grammar
  Turing machine

Part II. Algorithm Design Techniques:
  divide & conquer
  dynamic programming
  greedy
  graph algorithms

Part III. NP-Completeness (undecidability)
"3SUM problem"

Given $n$ numbers, do there exist 3 numbers summing to 100?

E.g. $81, 43, 95, 20, 32, 74, 25$

Brute force: $O(n^3)$ time

Smarter alg: $O(n^2)$ time.

Fastest?

OPEN $\sim O\left(\frac{n^2}{(\log n)^4}\right)$ [C'18]

---

Ex2  Given $n$ polygons & rectangle, can they be packed in rectangle?

No efficient algm believed to be possible ("NP-complete")

---

Ex3  Given $n$ polygons, can they tile the entire plane?

"Undecidable"

---

Part I. Models of Computation

Math Preliminaries

Strings  A string is a finite sequence of symbols from a finite set $S$
Strings

A string is a finite sequence of symbols from a finite set $\Sigma$

- Alphabet: strings over $\Sigma = \{0,1\}$
  - e.g. $1001$, $01$, $010$, $1$

Let $\varepsilon$ denote the empty string
Let $\Sigma^*$ denote $\{\text{all strings over } \Sigma\}$

Let $x, y$ be strings:

a) length $|x|$
   - e.g. $|1001| = 4$, $|\varepsilon| = 0$

b) concatenation $xy$
   - e.g. $x = 10$, $y = 0110$ $\Rightarrow$ $xy = 100110$
   - $(xy)z = x(yz)$
   - $xy \neq yx$
   - $|xy| = |x| + |y|$
   - $\varepsilon x = x\varepsilon = x$

\[c) \text{ i}^{\text{th}} \text{ power } \ x^i = xx \cdots x \quad i \text{ times}\]
   - e.g. $(010)^3 = 010010010$
   - $|x^i| = i |x|$
   - $x^i = x \cdot x^{i-1}$
   - $x^0 = \varepsilon$

\[d) x \text{ is a substring of } y \text{ if } y = wxz \text{ for some strings } w, z\]
   - (prefix if $w = \varepsilon$, suffix if $z = \varepsilon$)
e) other ops: $x^R =$ reverse of $x$
(can be defined recursively.

$$x^R = \begin{cases} e & \text{if } x = e \\ y^Ra & \text{if } x = ay \text{ for some } a \in \Sigma, y \in \Sigma^* \end{cases}$$

$$(xy)^R = y^Rx^R$$

languages

A language is a set of strings
i.e. $L \subseteq \Sigma^*$.

E.g. 
\{ 100, 01, 101, 0 \}\n\{ all words in English dictionary \} 
\{ all words in English dictionary \} 
\{ all syntactically valid programs in Python \} 
\{ all prime numbers written in binary \} 

Let $L_1, L_2$ be languages.

a) union $L_1 \cup L_2$
intersection $L_1 \cap L_2$
complement $\overline{L_1}$ ($= L_1^c$) ($= \Sigma^* \setminus L_1$)
difference $L_1 \setminus L_2 = L_1 \cap \overline{L_2}$

b) concatenation
$L_1L_2 = \{ xy : x \in L_1, y \in L_2 \}$

E.g. $L_1 = \{ 0, 00 \}$, $L_2 = \{ 1, 01 \}$
e.g. \( L_1 = \{0, 00\} \), \( L_2 = \{1, 01\} \)
\( L_1 L_2 = \{01, 0001, 001\} \)

e.g. \( L_1 = \{0, 00, 000, \ldots\} = \{0^i : i \geq 1\} \)
\( L_2 = \{1, 11, 111, \ldots\} = \{1^j : j \geq 1\} \)
\( L_1 L_2 = \{0^i 1^j : i, j \geq 1\} \).